IE School of Human Sciences & Technology

Modern Data Architectures I

Understanding Gaming Communities through Twitter Analysis

Group G

Student Names:

1. Maria Graciela Missura Donoso
2. Niklas Albus
3. Paul Ribes
4. Mario Hernan Fermin
5. Krishna Agrawal
6. Robert Ellis Bareuther

Course:

Master's in Business Analytics and Big Data

Table of Contents

Problem Description

Xbox and PlayStation. PlayStation and Xbox. They are the two most followed brands on Twitter, amassing over 50 million followers between the two of them. These followers represent more than just an army of likes and retweets – they represent a hyperactive online community that drives a large portion of the companies' revenues. Sony, the corporation behind the PlayStation gaming console, did $17.44 billion in gaming revenue their most recent fiscal year[1], while Microsoft, the manufacturer of the Xbox console, achieved a close second with $15.3 billion in gaming revenue[2]. As the industry leaders in a lucrative industry, every customer matters, especially when a customer is going to pay $500+ for a console, $100+ a year on subscriptions, and even more money on individual games.

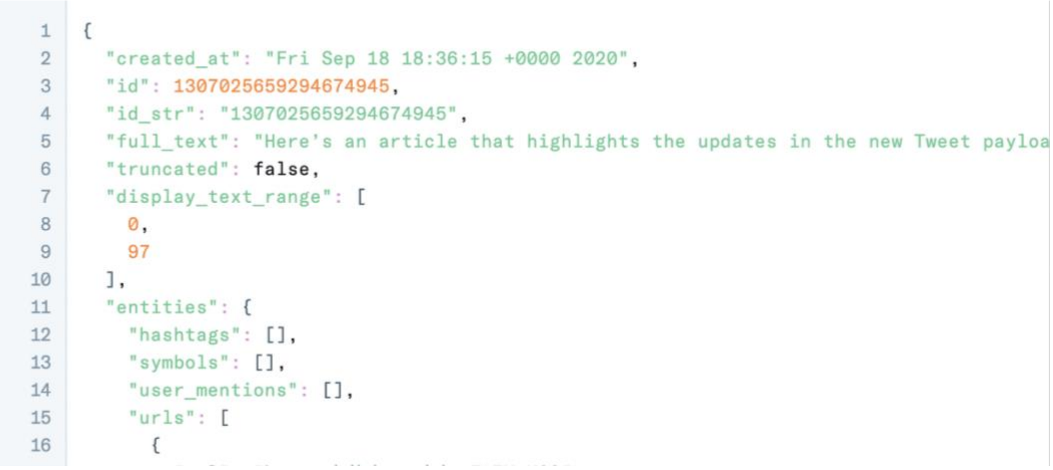
For many customers, the decision of which console to buy can be a very social one. Although PlayStation and Xbox try to differentiate themselves through technical innovations and exclusive offerings, what really matters to fans are the gaming communities that surround each platform. To the average gamer, it matters less which games are available on a certain console – they want to be playing on the same system as their real/online friends and the professionals/influencers they follow. This is why social media platforms, and Twitter specifically, are so important in gaining margin in the console gaming industry. Recognizing the value social media has, brands like PlayStation and Xbox are forced to invest huge amounts of money into social media marketing and content in order to grow their customer base.

In this project, we are performing an online community analysis on the tweets surrounding the PlayStation and Xbox brands. The Twitter data we collect will be used to analyze and evaluate how each brand is using Twitter to connect with their customers and foster an online community. Through our research and analysis, we are looking to understand how the Twitter gaming ecosystem functions – we want to understand not only how PlayStation and Twitter interacts with their customers and fans, but we also want to know how the fans interact with the brands, and how the fans behave with each other. With this information, we will have a holistic idea of which brand is performing better on Twitter, and therefore the marketplace.

[1] Sony Earnings Release FY20 Q4 https://www.sony.com/en/SonyInfo/IR/library/presen/er/pdf/20q4_sony.pdf
[2] Microsoft Earnings Release FY21 Q4 https://www.microsoft.com/en-us/Investor/earnings/FY-2021-Q4/segment-revenues

Data Sources

All data in this project is being received from Twitter. We accessed the Twitter data by making a Twitter Developer account, which allows us to pull data using a Twitter API based on certain parameters. Tweet data is stored as .json files, which allows Twitter to store valuable data regarding each tweet in a series of key-value pairs. Besides the actual content, tweets can have more than 150 attributes, including geolocation, user information, hashtags, and more[3]. JSON is the perfect format to store this kind of information because it allows for Twitter to create nested dictionaries sort the data of a tweet depending on its different built-in objects: the tweet itself, the user, geolocation, entities, and extended/additional entities. Below is an example of what the first 16 lines of a .json of a tweet might look like:



```
 1  {
 2    "created_at": "Fri Sep 18 18:36:15 +0000 2020",
 3    "id": 1307025659294674945,
 4    "id_str": "1307025659294674945",
 5    "full_text": "Here's an article that highlights the updates in the new Tweet payloa
 6    "truncated": false,
 7    "display_text_range": [
 8      0,
 9      97
10    ],
11    "entities": {
12      "hashtags": [],
13      "symbols": [],
14      "user_mentions": [],
15      "urls": [
16        {
```

https://developer.twitter.com/en/docs/twitter-api/v1/data-dictionary/object-model/example-payloads

You can see the use of nested dictionaries in this example, especially in line 11 where the object "entities" is created to store data such as hashtags, symbols used, and users mentioned. This is the power of Twitter Data, which takes something that seems to be a simple action like a short message and turns it into a goldmine for data analysis.

[3] Twitter - Data dictionary: Standard v1.1 https://developer.twitter.com/en/docs/twitter-api/v1/data-dictionary/overview

Data Ingestion

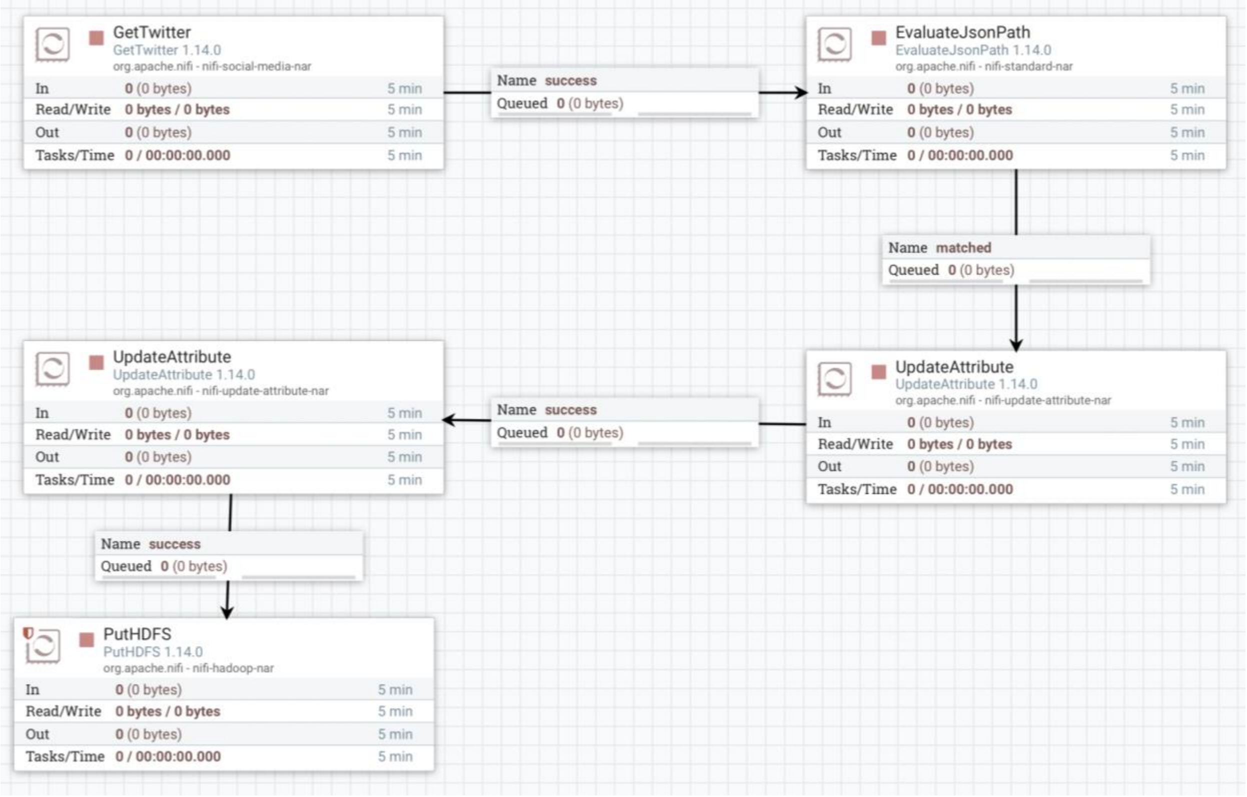
We are ingesting the Twitter data using a technology called Apache NiFi, which allows us to ingest streaming data from the Twitter API into the Hadoop Distributed File System (HDFS) at efficiently and at scale. NiFi works for both data in motion and data at rest, but for this project we will be using it to ingest data in motion as tweets are fed into NiFi in a live state. A large advantage of NiFi is its easy-to-understand routing pipelines diagrams, which show exactly what is happing to the data as it is being extracted, manipulated, and moved to its storage space.

The process for data ingestion was as follows:

1. We connected to our Virtual Machine (VirtualBox) using localhost and opened a terminal.
2. Using the terminal, we started Hadoop using the command: Hadoop-start.sh
3. Now that Hadoop is open, we started NiFi using the command: NiFi.sh start
4. Now that NiFi is active, we can enter the personal NiFi Web Interface.
5. Using the pipeline from the twitter/bitcoin lab, we changed the hashtags to filter for the tweets we want. In this investigation, we searched for tweets including "#ps5" for the first ingestion and "#xbox" for the second.
6. Before we could start the ingestion, we also needed to change the HDFSProcessor Path also to twitter/ps5 and twitter/Xbox to make sure the data gets where we want it to go.
7. Now that the path was ready, we could start ingestion.
8. We ingested a total of 40 mb of tweets (around 20mb for each brand), which we found to be sufficient, so we stopped the ingestion.
9. We saved the twitter data within our datalake/raw/twitter/… path

NiFi Data Pipelines Diagram

Below is the pipeline that NiFi took the data, starting with the extracting of the data through the Twitter API “GetTwitter”, then analyzing and evaluating the JSON path, updating the attributes to match the storage format, and finally load the data into the local HDFS.



GetTwitter
GetTwitter 1.14.0
org.apache.nifi - nifi-social-media-nar
In 0 (0 bytes) 5 min
Read/Write 0 bytes / 0 bytes 5 min
Out 0 (0 bytes) 5 min
Tasks/Time 0 / 00:00:00.000 5 min

Name success
Queued 0 (0 bytes)

EvaluateJsonPath
EvaluateJsonPath 1.14.0
org.apache.nifi - nifi-standard-nar
In 0 (0 bytes) 5 min
Read/Write 0 bytes / 0 bytes 5 min
Out 0 (0 bytes) 5 min
Tasks/Time 0 / 00:00:00.000 5 min

Name matched
Queued 0 (0 bytes)

UpdateAttribute
UpdateAttribute 1.14.0
org.apache.nifi - nifi-update-attribute-nar
In 0 (0 bytes) 5 min
Read/Write 0 bytes / 0 bytes 5 min
Out 0 (0 bytes) 5 min
Tasks/Time 0 / 00:00:00.000 5 min

Name success
Queued 0 (0 bytes)

UpdateAttribute
UpdateAttribute 1.14.0
org.apache.nifi - nifi-update-attribute-nar
In 0 (0 bytes) 5 min
Read/Write 0 bytes / 0 bytes 5 min
Out 0 (0 bytes) 5 min
Tasks/Time 0 / 00:00:00.000 5 min

Name success
Queued 0 (0 bytes)

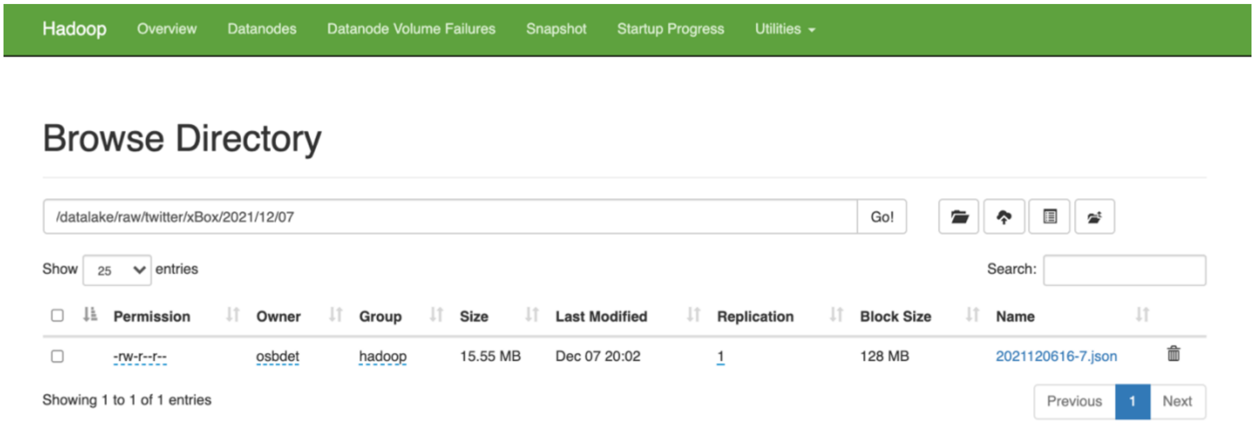
PutHDFS
PutHDFS 1.14.0
org.apache.nifi - nifi-hadoop-nar
In 0 (0 bytes) 5 min
Read/Write 0 bytes / 0 bytes 5 min
Out 0 (0 bytes) 5 min
Tasks/Time 0 / 00:00:00.000 5 min

Data Storage

Our data is being stored in a Hadoop Distributed File System (HDFS) built into our Virtual Machine and stored in our local data lake in .json format. Hadoop HDFS is a distributed file system, which means that it is a structured repository of files that are prepared to be extracted and analyzed by applications like Apache Spark and Hadoop. HDFS is a workhorse and is an industry standard, making it highly compatible with most data analysis programs. This is very convenient for us, as we will be analyzing this data through a combination of Apache Spark and Python: PySpark.

HDFS directory structure:

As seen below, our raw Twitter data for Xbox and PlayStation is stored in our local data lake within the Hadoop HDFS software. The file hierarchy is structured based on the state of the data (raw), the source of the data (Twitter), the brand the data is referring to (xBox/Ps5), and finally the date of the extraction (2021/12/07).

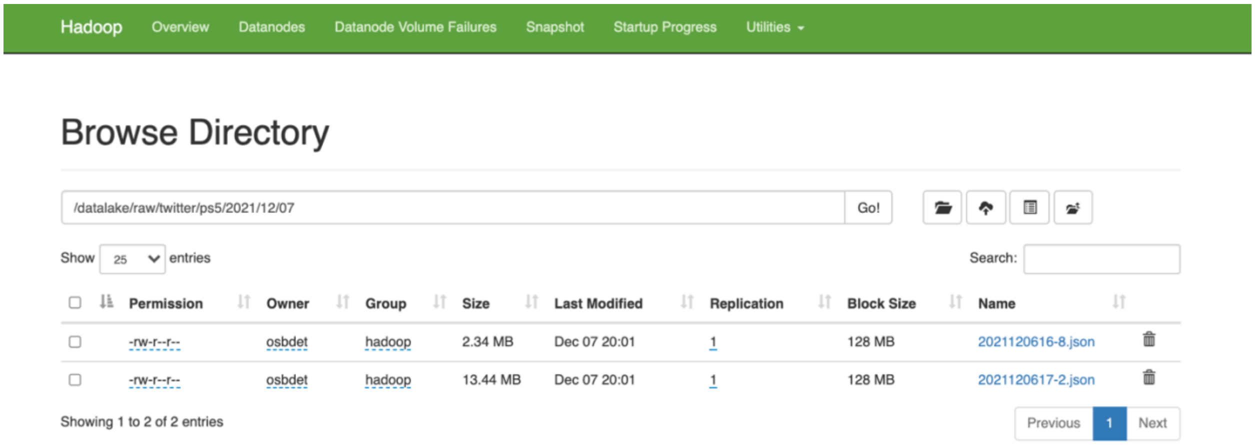


Browse Directory

/datalake/raw/twitter/xBox/2021/12/07

□	Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
□	-rw-r--r--	osbdet	hadoop	15.55 MB	Dec 07 20:02	1	128 MB	2021120616-7.json

Showing 1 to 1 of 1 entries



Browse Directory

/datalake/raw/twitter/ps5/2021/12/07

□	Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
□	-rw-r--r--	osbdet	hadoop	2.34 MB	Dec 07 20:01	1	128 MB	2021120616-8.json
□	-rw-r--r--	osbdet	hadoop	13.44 MB	Dec 07 20:01	1	128 MB	2021120617-2.json

Showing 1 to 2 of 2 entries

Data Processing

We are processing our data using two technologies: Apache Spark and Python, which together are called PySpark. PySpark is a powerful data analysis tool because it takes the strength of Apache Spark's processing power and scalability and pairs it with Python's power in data analysis, giving it the power to control Spark[4]. This allows us to use many of Spark's APIs within a Python Jupyter Notebook, including Spark SQL, a powerful tool for querying and analyzing data, pairing it with Spark's Pandas library to create robust DataFrames. Screenshots

[4] PySpark Documentation https://spark.apache.org/docs/latest/api/python/

of our Spark notebooks can be found attached to this document in the appendix, as well as the original .ipynb files attached to the assignment.

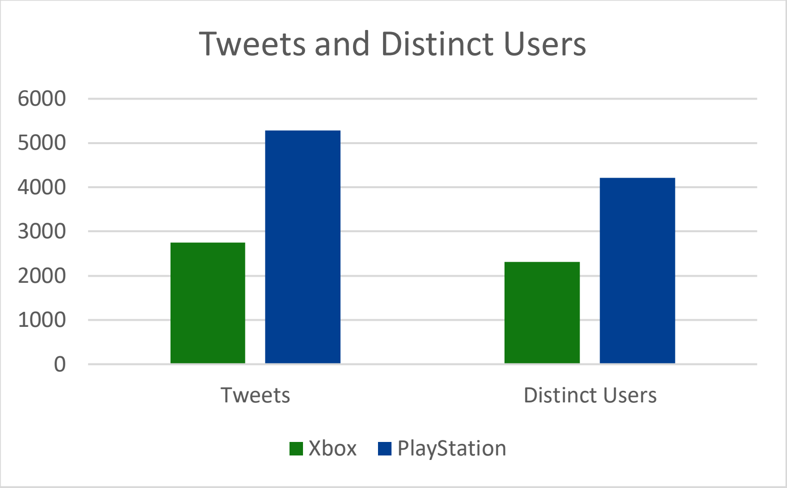
Our data processing was as follows:

1. We started our Virtual Machine (Headless Start)
2. We went into our browser and type http://localhost:28888 to access our Jupyter Notebooks
3. We opened our Terminal
4. We started Hadoop using the command: Hadoop-start.sh
5. We uploaded the two notebooks from twitter to Jupyter
6. First, we opened the "Raw to Std" notebook
7. In another tab, we made sure to open: http://localhost:50070/explorer.html#/datalake/ to define the data path in Hadoop HDFS.
8. In our Hadoop directory, we placed the exact path as in the notebook. (datalake/raw /twitter/ps5/2021/12/07)
9. We changed the paths within the notebook to the one created in Hadoop (first Ps5 and then xBox)
10. We ran the notebook for ps5 first and then xBox
11. The notebook created parquets files and the std data in the Hadoop Directory that can now be analyzed
12. We ran the second notebook Twitter-Analytics to draw conclusions from the std data

Project Results & Analysis

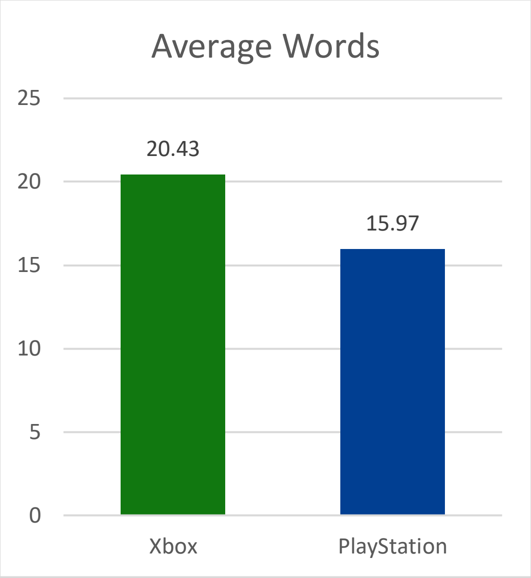
Tweets and Users:

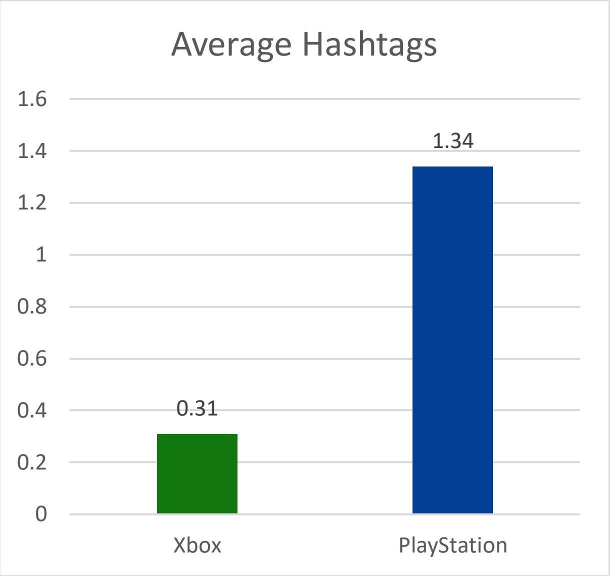
Calculated by applying the .count() function, we found that Ps5 is getting mentioned nearly twice as much as Xbox in our dataset (5280 tweets vs 2754 tweets). Using .distinct(), we discovered that the number of distinct users is less than the number of posts (4209 and 2307 respectively), which concludes that some people made multiple tweets about the consoles (1.25 and 1.19 tweets/account). We found this to be a surprising difference, given how similar he followership of both brands is much closer than this gap in tweets might suggest. Looking at the yearly revenue for Xbox and PlayStation does not provide a clear answer either – their global revenues are also considerably similar. Considering that PlayStation and Xbox are very comparable in terms of sales and following, the answer must come down to a difference between the online communities and how these communities interact with the brands.



<u>Hashtags and Original Content</u>:

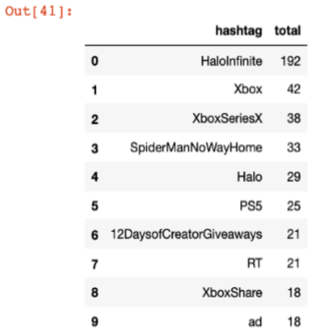
The next criteria we evaluated for is the maximum and average number of hashtags appearing in our dataset, as well as the number of words used per tweet. Querying the entity "hashtags" and calculating the average, we found that less hashtags being used within Xbox posts, with an average of 0.312 hashtags rather than PlayStation's 1.34. This means that PlayStation tweets contain, on average, three times as many hashtags. On the other hand, posts related to Xbox seemed to be longer as the average number of words was larger, with the average Xbox tweet containing 20.43 words in comparison to PlayStation's 15.97 words. While the PlayStation twitter community might be more trend-focused, the Xbox community appears more actively engaged, albeit smaller. Furthermore, this suggests that PlayStation is the better marketer of the two, as they seem more able to engage their followers through hashtags, which generate more engagement and visibility, which might explain their larger presence on Twitter.





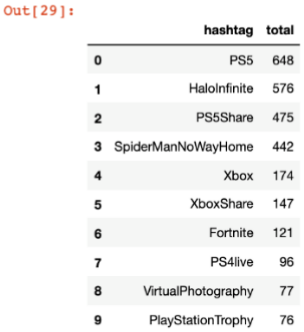
The next search criteria was to look for the top 10 more popular Hashtags. Here, we may see that tweets for Xbox have a most popular Hashtag being "HaloInfinite", which is a new and incredibly popular game being released exclusively for Xbox. On the other side, "PlayStation" is mentioned the most for Ps5 and their second most popular is also "HaloInfinite", meaning that Xbox's decision to make Halo an exclusive game is heavily effecting PlayStation fans.

Xbox Hashtags:


Out[41]:

	hashtag	total
0	HaloInfinite	192
1	Xbox	42
2	XboxSeriesX	38
3	SpiderManNoWayHome	33
4	Halo	29
5	PS5	25
6	12DaysofCreatorGiveaways	21
7	RT	21
8	XboxShare	18
9	ad	18

PlayStation Hashtags:


Out[29]:

	hashtag	total
0	PS5	648
1	HaloInfinite	576
2	PS5Share	475
3	SpiderManNoWayHome	442
4	Xbox	174
5	XboxShare	147
6	Fortnite	121
7	PS4live	96
8	VirtualPhotography	77
9	PlayStationTrophy	76

Users and Influencers:

In this section of our analysis, we decided to find the most important individuals and accounts on Twitter who are driving engagement in the online gaming ecosystem. We found these users by using the following filtering criteria: most mentions, most followers, and most tweets. All of the users were identified using .select() function, grouping by each criteria we were looking to sort for.

The user data returns a valuable and varied set of important accounts that help us understand the online gaming communities that surround the two brands we are investigating. Across the three criteria we are scanning, we can find three types of users: independent gaming influencers and enthusiasts, related and complementary brands, and finally, accounts managed directly by Xbox and PlayStation.

Top Xbox and PlayStation Twitter Users (mentions, follower count, tweets posted)

Brand	Number	User	Mentions	User	Followers (thousands)	User	Tweets (thousands)
Xbox	1	mkbhd	1075	behzinga	2550	Xboxsupport	3008
	2	mattswider	164	xboxsupport	1798	streamer_boost	1552
	3	xbox	124	jhonnycharles88	258	helperstream	587
	4	halo	120	godfreycomedian	129	elfyau	470
PlayStation	1	playstation	323	playstation	23178	sectest9	2520
	2	halo	256	blizzardcs	1016	streamerwall	2164
	3	insomniacgames	196	ella_exclusive	673	ronniehowlett3	1668
	4	spiderman	178	chaosxsilencer	440	streamer_boost	1552

For Xbox, influencers like Behzinga, Mkbhd, and Jhonnycharles98. For PlayStation, these users would be Ella_exclusive and Chaosxsilencer. These users represent the diversity of twitter, as all occupy several spaces – some are popular influencers that mainly focus on other sectors but share their passion for Xbox or PlayStation, while others are actual focused gaming influencers that push the culture forward. Either way, these accounts are incredibly important to these brands because they help customers form personal connections to the brands, whether the content is directly targeted or not.

Beyond the influencers and independent users, brand accounts are incredibly important to gaming discourse on Twitter. These accounts announce breaking news (new games, updates, etc), they interact directly and indirectly with fans, and are used to conduct mass-marketing campaigns. While complementary accounts like Halo, Marvel, Blizzard, and Spiderman appear for both brands, it is also important to note that accounts like XboxSupport and AskPS_UK appear in the analysis, as these types of accounts allow brands to interact directly with users rapidly and at scale, resolving their doubts in a public forum and establishing their presence.

Geolocation:

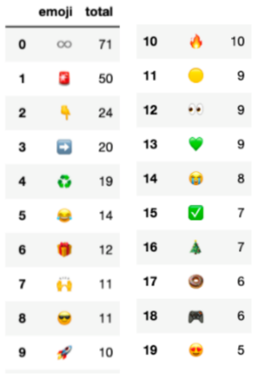
We also analyzed the geolocation of the tweets to see which top three countries are tweeting about each of the brands. We did this by grouping by the attribute "place.country" and ordering by "tweets_posted". While the majority of the tweets did not have a geolocation, after eliminating this data we were able to analyze the top countries that appeared. This reveals some incredibly interesting data – mainly that the Xbox audience is much more heavily skewed towards the United States while the PlayStation audience is more international, with the UK, the US, and India appearing with relatively balanced percentages. This might explain PlayStation's larger presence on Twitter, as their fanbase is more global and therefore larger.

Brand	Country	Tweets	%/Total
Xbox	United States	766507	96.23%
	United Kingdom	27266	3.42%
	India	2790	0.35%
PlayStation	United Kingdom	31948	50.21%
	United States	18753	29.47%
	India	12928	20.32%

Emojis:

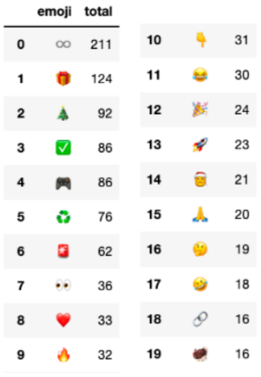
Lastly, we scanned the data to find the most popular emoticons used in conjunction with each brand. This was a more complicated case, as we needed to import an emoji set into our terminal and then import it into the PySpark session. Once there, we could use the library to create a list of emojis and use it to group the Twitter data. For both cases we are able to see that the emoji "glasses" comes up the most. However, Ps5 has more than three times the amount of hashtags being used compared to Xbox. Underlining, Ps5 second and third most used emojis are a present and a Christmas tree. The present emoji occur ten times more than for PlayStation than for Xbox. This bodes well for PlayStation in the holiday season, as there is more excitement around gift giving and consumption.

Xbox:



	emoji	total
0	♾	71
1	🚨	50
2	👇	24
3	➡️	20
4	♻️	19
5	😂	14
6	🎁	12
7	🙌	11
8	😎	11
9	🚀	10
10	🔥	10
11	🟡	9
12	👀	9
13	💚	9
14	😭	8
15	✅	7
16	🎄	7
17	🍩	6
18	🎮	6
19	😍	5

PlayStation:



	emoji	total
0	♾	211
1	🎁	124
2	🎄	92
3	✅	86
4	🎮	86
5	♻️	76
6	🚨	62
7	👀	36
8	❤️	33
9	🔥	32
10	👇	31
11	😂	30
12	🎉	24
13	🚀	23
14	🎅	21
15	🙏	20
16	🤔	19
17	🤣	18
18	🔗	16
19	🦔	16

Conclusions

After a deep-dive of the data, we can conclude that PlayStation's twitter gaming community is certainly the stronger of the two. The composition of PlayStation's community is more international, is experiences more brand interaction from fans (with both PS-run accounts and related brand accounts, and lastly, experiences more engagement through hashtags and other viral factors. Looking at the current trends of Christmas excitement through emojis and a larger volume of tweets, it is clear that PlayStation is implementing a stronger social media strategy than Xbox. This is not to say that Xbox is doing poorly, though, as they have also done well to create an online ecosystem and gaming community. Xbox's strengths on Twitter may not lie in their viral-ness or volume of tweets, but they do have a higher quality of content that PlayStation cannot match. This comes through in their powerful influencers, such as Behzinga and Mkbhd, who promote the brand to their millions of combined followers. This is also reflected in the tweet length, as Xbox tweets tend to contain more original content rather than hashtags. Taking all of this into account, the two brands have a lot to learn from each other – PlayStation must invest more in the quality of their online community's content, while Xbox must research how to make their content more viral so that it can match PlayStation's volume.

References

Sony Earnings Release FY20 Q4

https://www.sony.com/en/SonyInfo/IR/library/presen/er/pdf/20q4_sony.pdf

Microsoft Earnings Release FY21 Q4

https://www.microsoft.com/en-us/Investor/earnings/FY-2021-Q4/segment-revenues

Twitter - Data dictionary: Standard v1.1

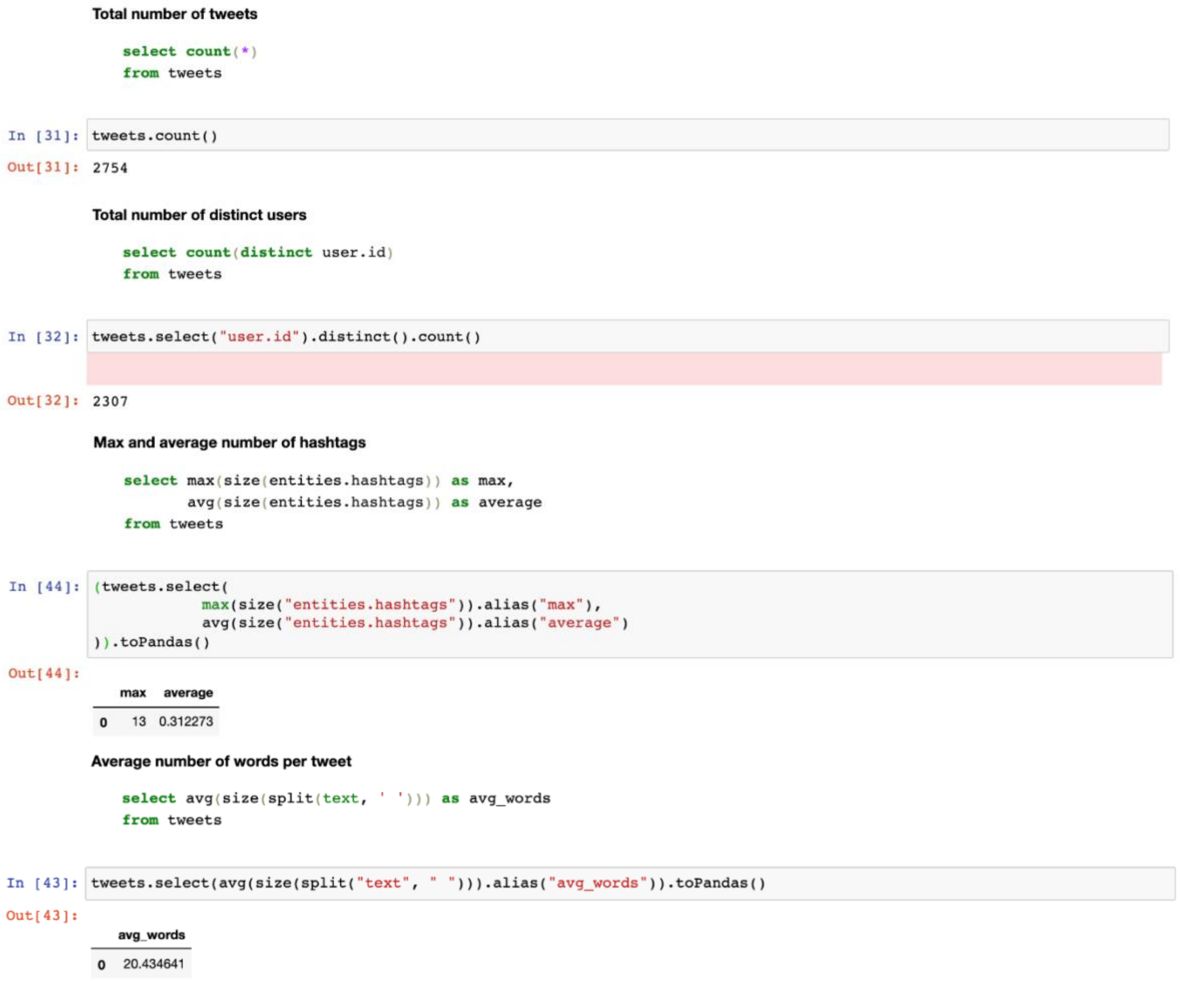
https://developer.twitter.com/en/docs/twitter-api/v1/data-dictionary/overview

PySpark Documentation

https://spark.apache.org/docs/latest/api/python/

Appendix

Xbox



Total number of tweets

```
select count(*)
from tweets
```

```
In [31]: tweets.count()
Out[31]: 2754
```

Total number of distinct users

```
select count(distinct user.id)
from tweets
```

```
In [32]: tweets.select("user.id").distinct().count()

Out[32]: 2307
```

Max and average number of hashtags

```
select max(size(entities.hashtags)) as max,
       avg(size(entities.hashtags)) as average
from tweets
```

```
In [44]: (tweets.select(
                     max(size("entities.hashtags")).alias("max"),
                     avg(size("entities.hashtags")).alias("average")
         )).toPandas()
Out[44]:
```

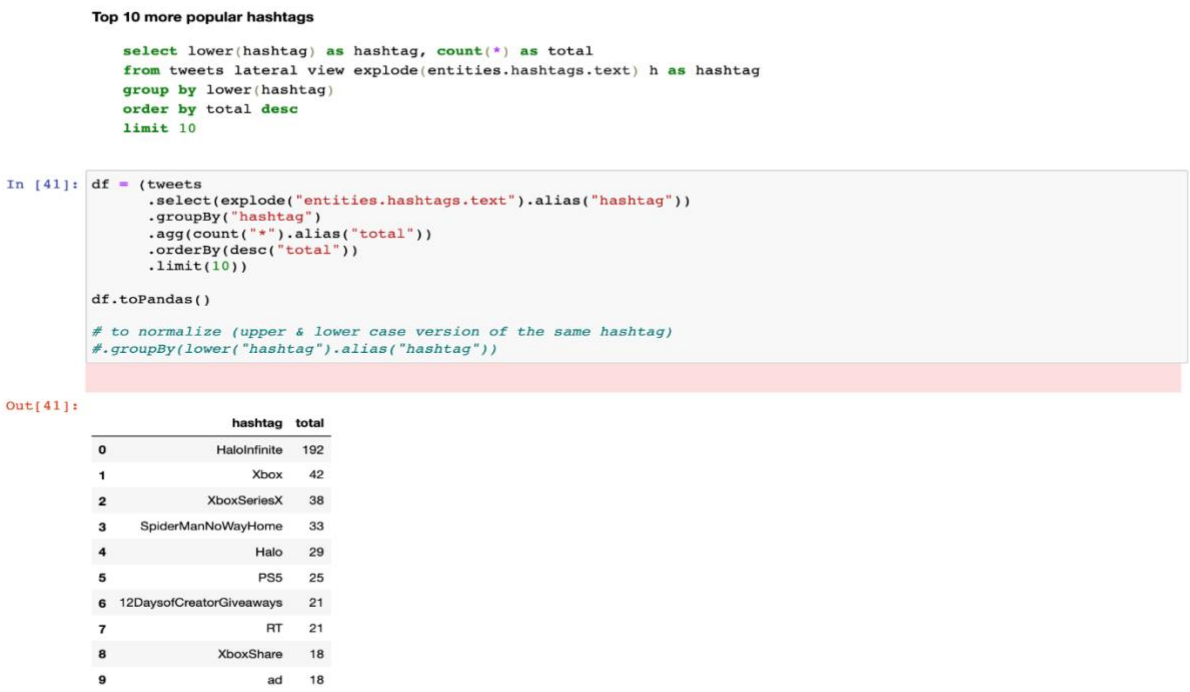
	max	average
0	13	0.312273

Average number of words per tweet

```
select avg(size(split(text, ' '))) as avg_words
from tweets
```

```
In [43]: tweets.select(avg(size(split("text", " "))).alias("avg_words")).toPandas()
Out[43]:
```

	avg_words
0	20.434641



Top 10 more popular hashtags

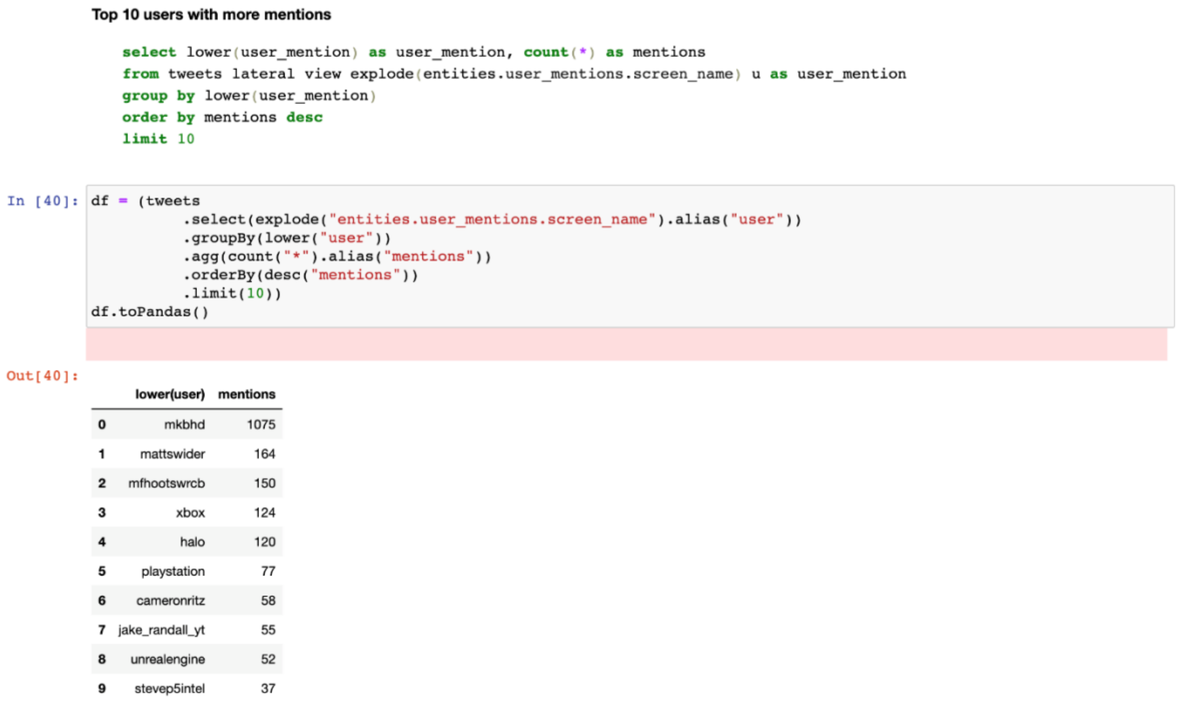
```
select lower(hashtag) as hashtag, count(*) as total
from tweets lateral view explode(entities.hashtags.text) h as hashtag
group by lower(hashtag)
order by total desc
limit 10
```

In [41]:

```
df = (tweets
      .select(explode("entities.hashtags.text").alias("hashtag"))
      .groupBy("hashtag")
      .agg(count("*").alias("total"))
      .orderBy(desc("total"))
      .limit(10))

df.toPandas()

# to normalize (upper & lower case version of the same hashtag)
#.groupBy(lower("hashtag").alias("hashtag"))
```

Out[41]:

	hashtag	total
0	HaloInfinite	192
1	Xbox	42
2	XboxSeriesX	38
3	SpiderManNoWayHome	33
4	Halo	29
5	PS5	25
6	12DaysofCreatorGiveaways	21
7	RT	21
8	XboxShare	18
9	ad	18



Top 10 users with more mentions

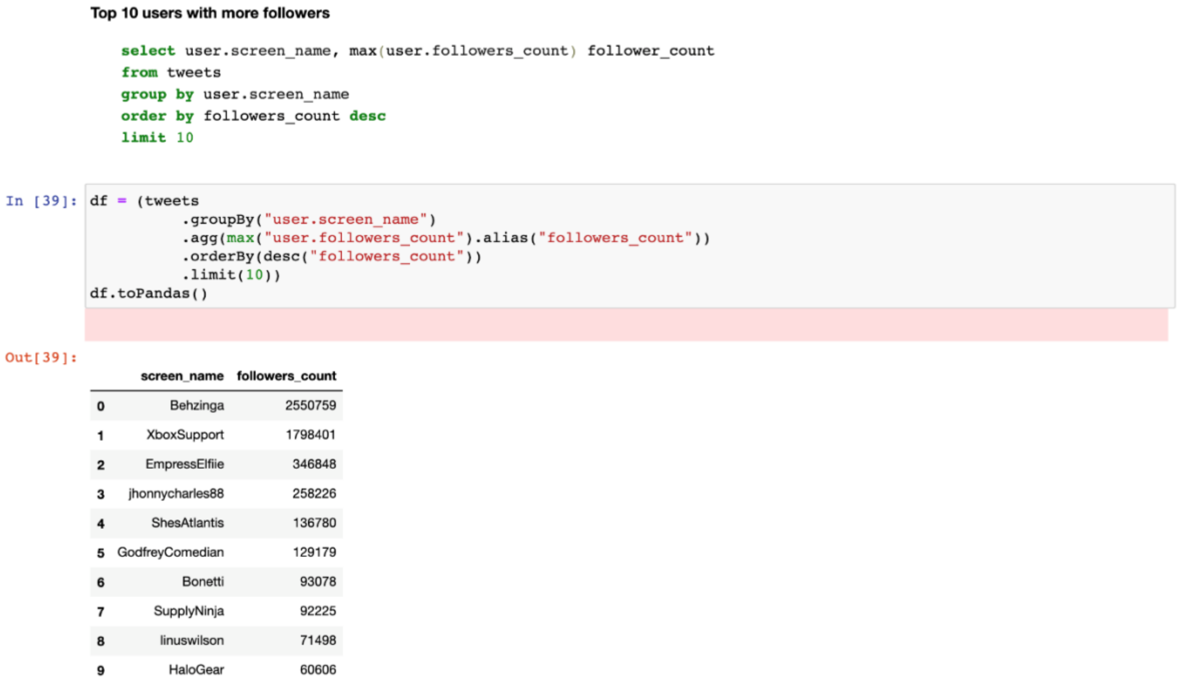
```
select lower(user_mention) as user_mention, count(*) as mentions
from tweets lateral view explode(entities.user_mentions.screen_name) u as user_mention
group by lower(user_mention)
order by mentions desc
limit 10
```

In [40]:

```
df = (tweets
            .select(explode("entities.user_mentions.screen_name").alias("user"))
            .groupBy(lower("user"))
            .agg(count("*").alias("mentions"))
            .orderBy(desc("mentions"))
            .limit(10))
df.toPandas()
```

Out[40]:

	lower(user)	mentions
0	mkbhd	1075
1	mattswider	164
2	mfhootswrcb	150
3	xbox	124
4	halo	120
5	playstation	77
6	cameronritz	58
7	jake_randall_yt	55
8	unrealengine	52
9	stevep5intel	37



Top 10 users with more followers

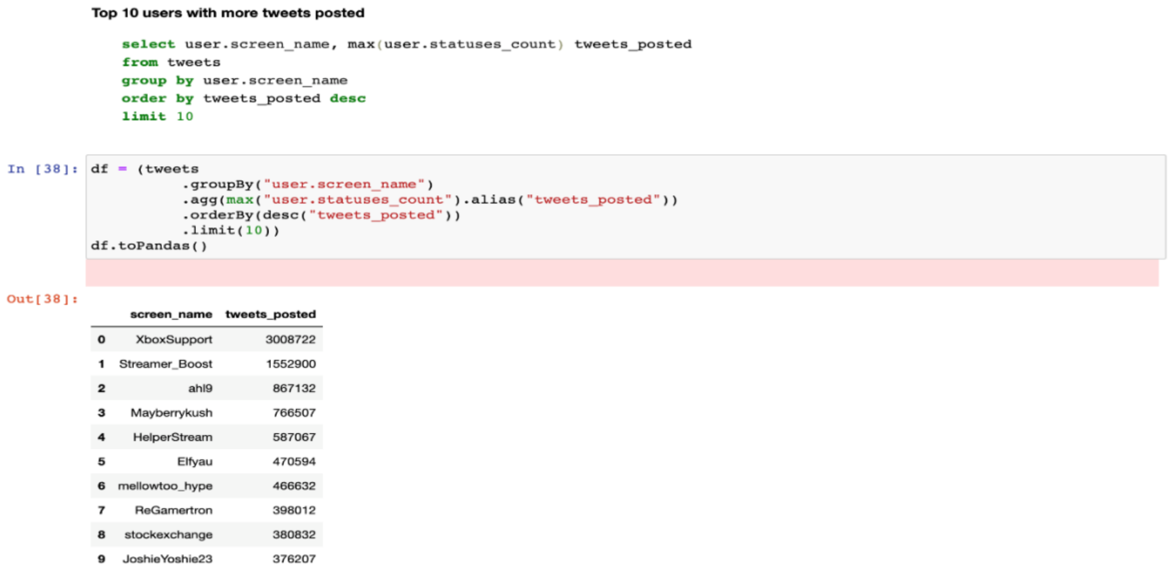
```
select user.screen_name, max(user.followers_count) follower_count
from tweets
group by user.screen_name
order by followers_count desc
limit 10
```

In [39]:

```
df = (tweets
          .groupBy("user.screen_name")
          .agg(max("user.followers_count").alias("followers_count"))
          .orderBy(desc("followers_count"))
          .limit(10))
df.toPandas()
```

Out[39]:

	screen_name	followers_count
0	Behzinga	2550759
1	XboxSupport	1798401
2	EmpressElfiie	346848
3	jhonnycharles88	258226
4	ShesAtlantis	136780
5	GodfreyComedian	129179
6	Bonetti	93078
7	SupplyNinja	92225
8	linuswilson	71498
9	HaloGear	60606



Top 10 users with more tweets posted

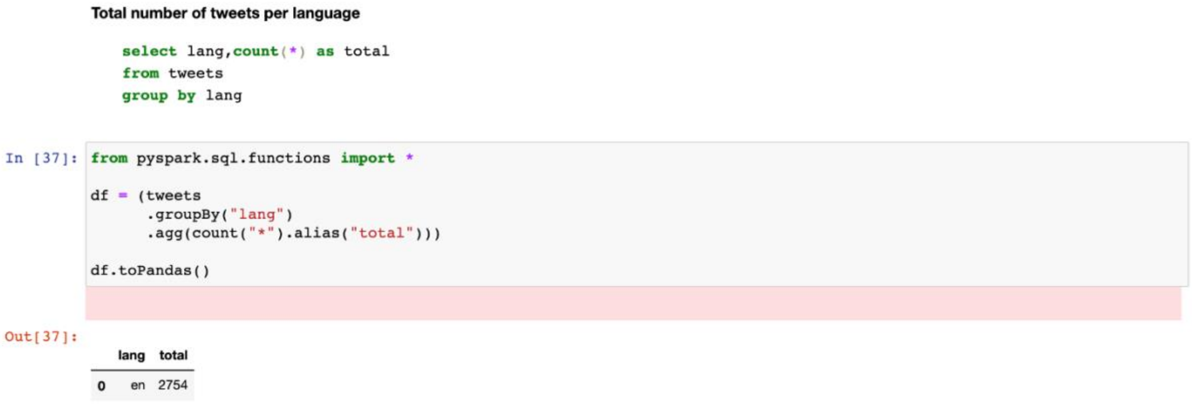
```
select user.screen_name, max(user.statuses_count) tweets_posted
from tweets
group by user.screen_name
order by tweets_posted desc
limit 10
```

In [38]:

```
df = (tweets
          .groupBy("user.screen_name")
          .agg(max("user.statuses_count").alias("tweets_posted"))
          .orderBy(desc("tweets_posted"))
          .limit(10))
df.toPandas()
```

Out[38]:

	screen_name	tweets_posted
0	XboxSupport	3008722
1	Streamer_Boost	1552900
2	ahl9	867132
3	Mayberrykush	766507
4	HelperStream	587067
5	Elfyau	470594
6	mellowtoo_hype	466632
7	ReGamertron	398012
8	stockexchange	380832
9	JoshieYoshie23	376207



Total number of tweets per language

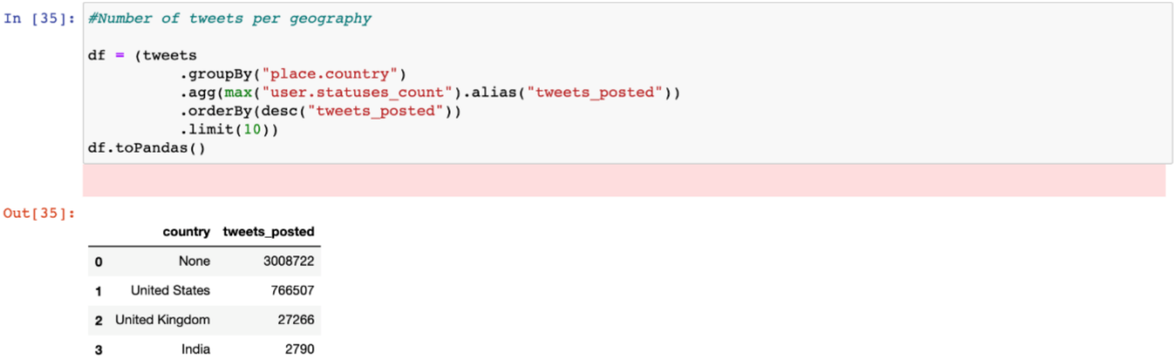
```
select lang,count(*) as total
from tweets
group by lang
```

In [37]:

```
from pyspark.sql.functions import *

df = (tweets
      .groupBy("lang")
      .agg(count("*").alias("total")))

df.toPandas()
```

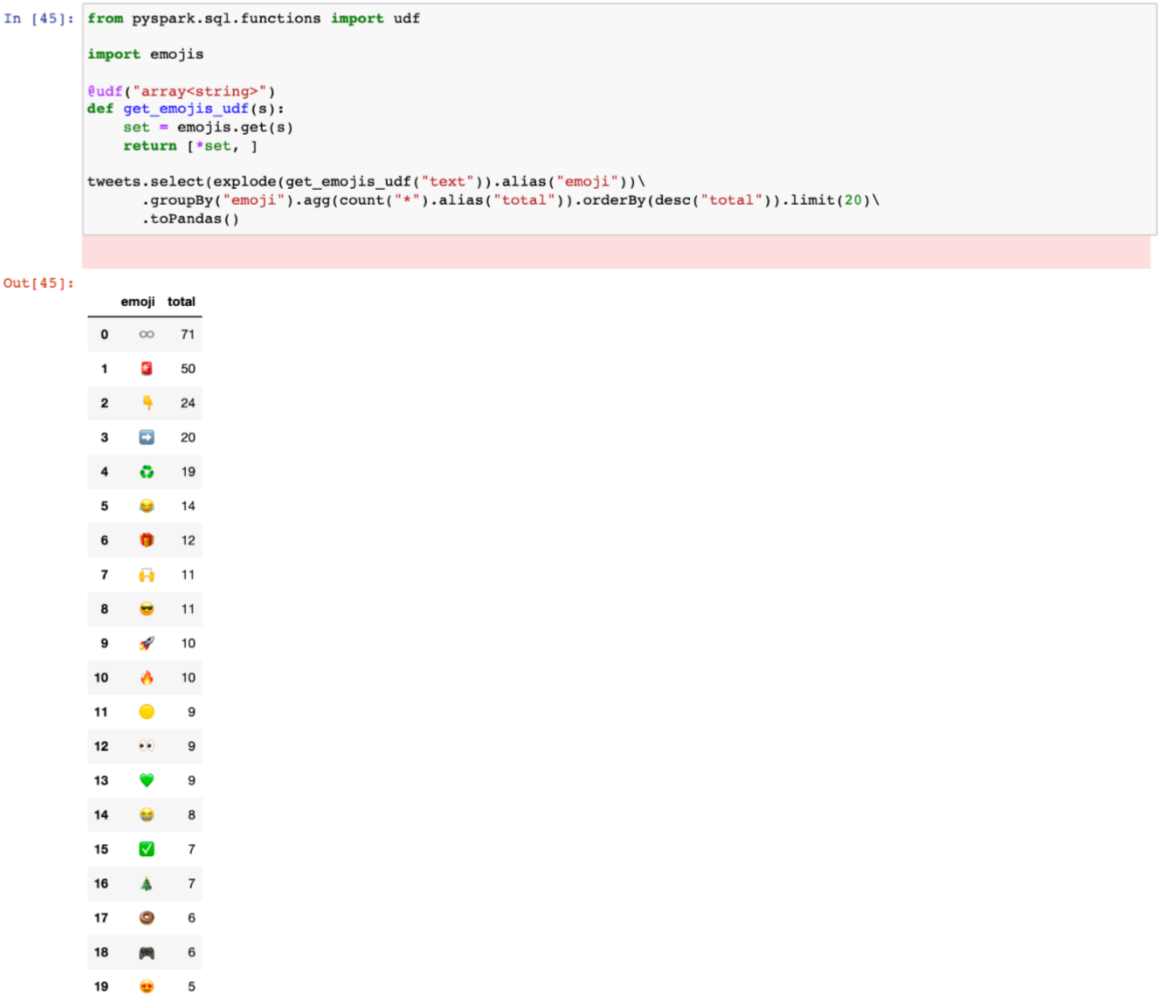
Out[37]:

	lang	total
0	en	2754



```
In [35]: #Number of tweets per geography

         df = (tweets
                   .groupBy("place.country")
                   .agg(max("user.statuses_count").alias("tweets_posted"))
                   .orderBy(desc("tweets_posted"))
                   .limit(10))
         df.toPandas()
```

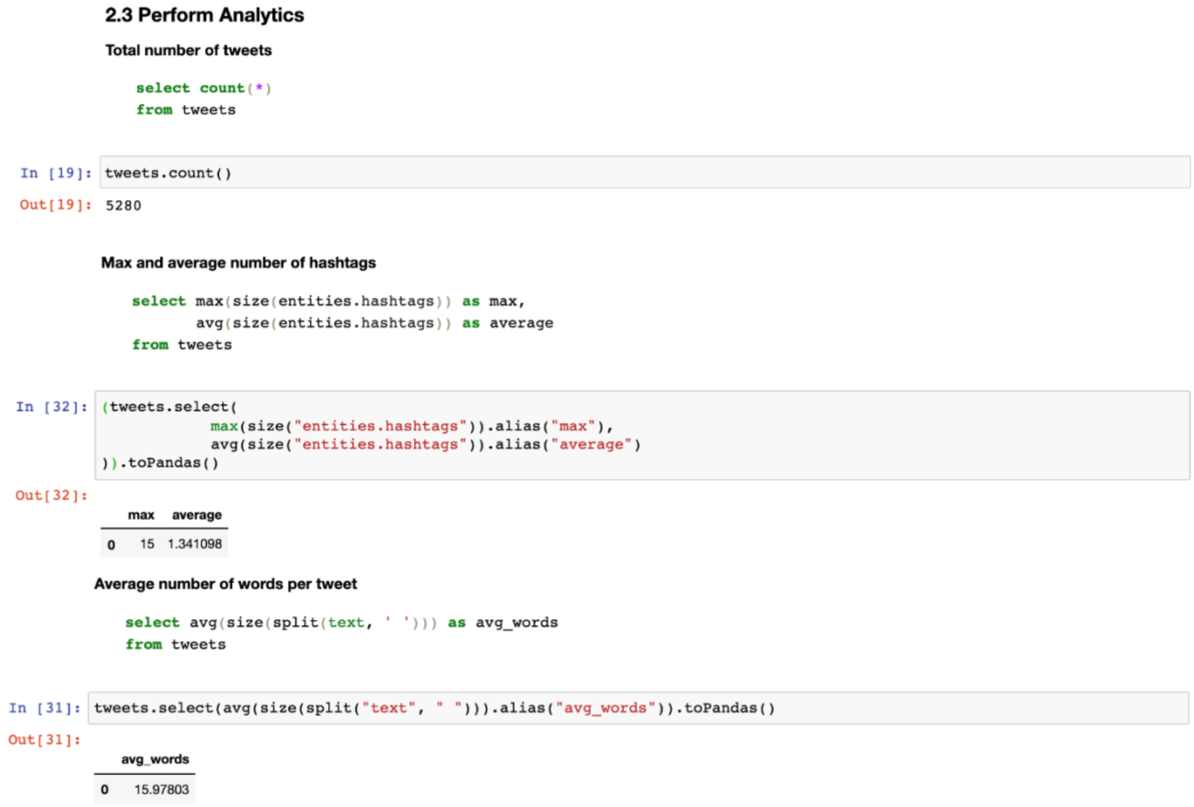
```
Out[35]:
                  country  tweets_posted
         0           None        3008722
         1  United States         766507
         2 United Kingdom          27266
         3          India           2790
```



```
In [45]: from pyspark.sql.functions import udf

         import emojis

         @udf("array<string>")
         def get_emojis_udf(s):
             set = emojis.get(s)
             return [*set, ]

         tweets.select(explode(get_emojis_udf("text")).alias("emoji"))\
               .groupBy("emoji").agg(count("*").alias("total")).orderBy(desc("total")).limit(20)\
               .toPandas()
```

```
Out[45]:
              emoji  total
          0     ♾️      71
          1     🚨      50
          2     👇      24
          3     ➡️      20
          4     ♻️      19
          5     😂      14
          6     🎁      12
          7     🙌      11
          8     😎      11
          9     🚀      10
         10     🔥      10
         11     🟡       9
         12     👀       9
         13     💚       9
         14     😭       8
         15     ✅       7
         16     🎄       7
         17     🍩       6
         18     🎮       6
         19     😍       5
```

Ps 5:



2.3 Perform Analytics

Total number of tweets

```
select count(*)
from tweets
```

```
In [19]: tweets.count()
Out[19]: 5280
```

Max and average number of hashtags

```
select max(size(entities.hashtags)) as max,
       avg(size(entities.hashtags)) as average
from tweets
```

```
In [32]: (tweets.select(
                     max(size("entities.hashtags")).alias("max"),
                     avg(size("entities.hashtags")).alias("average")
         )).toPandas()
Out[32]:
```

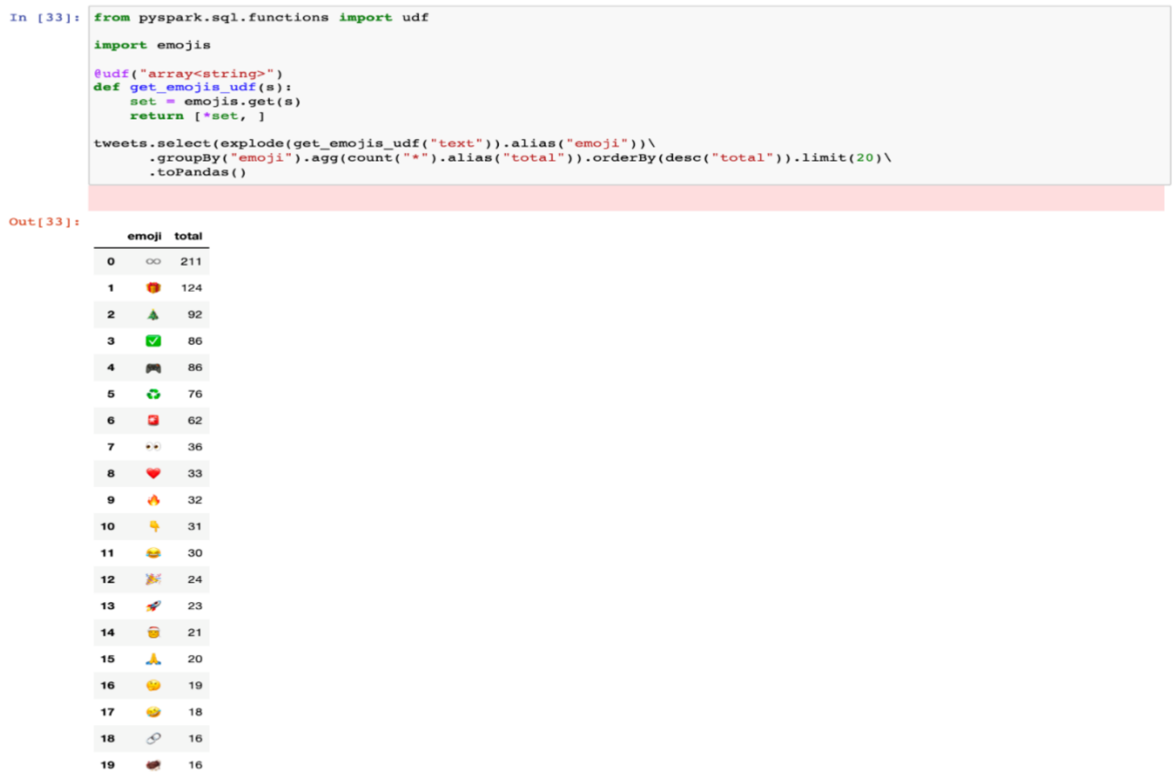
	max	average
0	15	1.341098

Average number of words per tweet

```
select avg(size(split(text, ' '))) as avg_words
from tweets
```

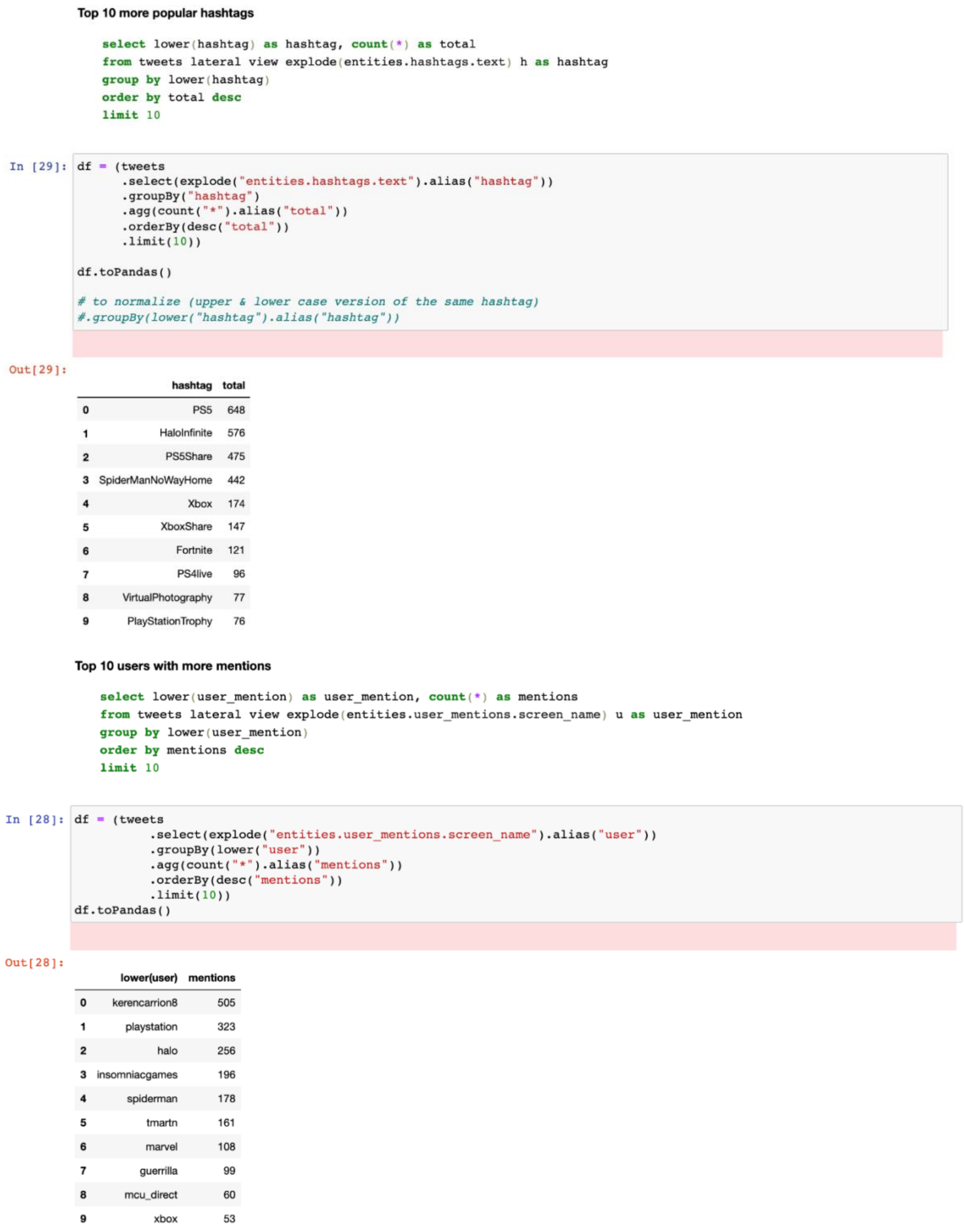
```
In [31]: tweets.select(avg(size(split("text", " "))).alias("avg_words")).toPandas()
Out[31]:
```

	avg_words
0	15.97803



```
In [33]: from pyspark.sql.functions import udf

         import emojis

         @udf("array<string>")
         def get_emojis_udf(s):
             set = emojis.get(s)
             return [*set, ]

         tweets.select(explode(get_emojis_udf("text")).alias("emoji"))\
               .groupBy("emoji").agg(count("*").alias("total")).orderBy(desc("total")).limit(20)\
               .toPandas()
Out[33]:
```

	emoji	total
0	♾	211
1	🎁	124
2	🎄	92
3	✅	86
4	🎮	86
5	♻️	76
6	🚨	62
7	👀	36
8	❤️	33
9	🔥	32
10	👇	31
11	😂	30
12	🎉	24
13	🚀	23
14	🎅	21
15	🙏	20
16	🤔	19
17	🤣	18
18	🔗	16
19	🍫	16



Top 10 more popular hashtags

```
select lower(hashtag) as hashtag, count(*) as total
from tweets lateral view explode(entities.hashtags.text) h as hashtag
group by lower(hashtag)
order by total desc
limit 10
```

In [29]:
```
df = (tweets
      .select(explode("entities.hashtags.text").alias("hashtag"))
      .groupBy("hashtag")
      .agg(count("*").alias("total"))
      .orderBy(desc("total"))
      .limit(10))

df.toPandas()

# to normalize (upper & lower case version of the same hashtag)
#.groupBy(lower("hashtag").alias("hashtag"))
```

Out[29]:

	hashtag	total
0	PS5	648
1	HaloInfinite	576
2	PS5Share	475
3	SpiderManNoWayHome	442
4	Xbox	174
5	XboxShare	147
6	Fortnite	121
7	PS4live	96
8	VirtualPhotography	77
9	PlayStationTrophy	76

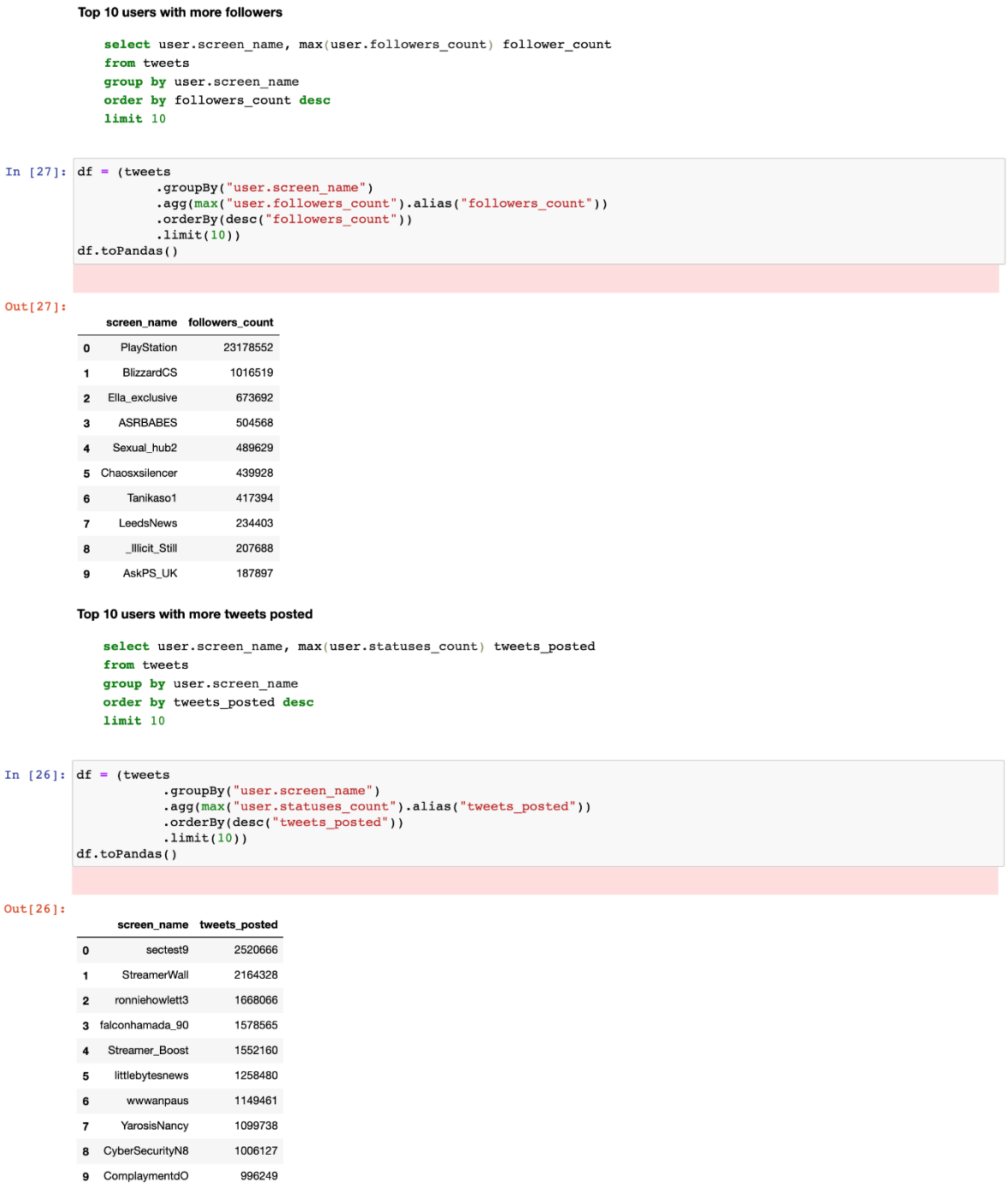
Top 10 users with more mentions

```
select lower(user_mention) as user_mention, count(*) as mentions
from tweets lateral view explode(entities.user_mentions.screen_name) u as user_mention
group by lower(user_mention)
order by mentions desc
limit 10
```

In [28]:
```
df = (tweets
          .select(explode("entities.user_mentions.screen_name").alias("user"))
          .groupBy(lower("user"))
          .agg(count("*").alias("mentions"))
          .orderBy(desc("mentions"))
          .limit(10))
df.toPandas()
```

Out[28]:

	lower(user)	mentions
0	kerencarrion8	505
1	playstation	323
2	halo	256
3	insomniacgames	196
4	spiderman	178
5	tmartn	161
6	marvel	108
7	guerrilla	99
8	mcu_direct	60
9	xbox	53



Top 10 users with more followers

```
select user.screen_name, max(user.followers_count) follower_count
from tweets
group by user.screen_name
order by followers_count desc
limit 10
```

In [27]:

```
df = (tweets
          .groupBy("user.screen_name")
          .agg(max("user.followers_count").alias("followers_count"))
          .orderBy(desc("followers_count"))
          .limit(10))
df.toPandas()
```

Out[27]:

	screen_name	followers_count
0	PlayStation	23178552
1	BlizzardCS	1016519
2	Ella_exclusive	673692
3	ASRBABES	504568
4	Sexual_hub2	489629
5	Chaosxsilencer	439928
6	Tanikaso1	417394
7	LeedsNews	234403
8	_Illicit_Still	207688
9	AskPS_UK	187897

Top 10 users with more tweets posted

```
select user.screen_name, max(user.statuses_count) tweets_posted
from tweets
group by user.screen_name
order by tweets_posted desc
limit 10
```

In [26]:

```
df = (tweets
           .groupBy("user.screen_name")
           .agg(max("user.statuses_count").alias("tweets_posted"))
           .orderBy(desc("tweets_posted"))
           .limit(10))
df.toPandas()
```

Out[26]:

	screen_name	tweets_posted
0	sectest9	2520666
1	StreamerWall	2164328
2	ronniehowlett3	1668066
3	falconhamada_90	1578565
4	Streamer_Boost	1552160
5	littlebytesnews	1258480
6	wwwanpaus	1149461
7	YarosisNancy	1099738
8	CyberSecurityN8	1006127
9	ComplaymentdO	996249